1 [2 puntos] Responda *razonadamente* si son ciertas las siguientes afirmaciones sobre el sistema de E/S:

a) Los módulos de E/S son un elemento hardware que facilita que cualquier periférico pueda ser manejado por cualquier CPU (arquitectura).

b) La E/S programada o directa sólo necesita de la CPU (arquitectura) que disponga de instrucciones *especiales* para escribir o leer de los registros de los módulos de E/S.

c) La duración de una operación de E/S depende sustancialmente de la capacidad de ejecución de la CPU (número de MIPS).

d) Las duración de una operación de E/S depende sustancialmente de la técnica de E/S empleada (directa, mediante interrupciones y por DMA).

SOLUCIÓN

a) Es una afirmación cierta. Los módulos de E/S se añadieron a los periféricos para evitar que la Unidad de Control (UC) tuviese que entender y gestionar las señales de estado y de control específicas de cada periférico. El módulo de E/S se comporta como una especie de pequeña UC que decodifica los mandatos que se graban en su registro de control y que codifica, de alguna forma, el estado del periférico en su registro de estado. También facilita la función imprescindible de *buffering* de datos entre el sistema y el periférico, a la vez que puede suministrar algunas funciones auxiliares como la detección de errores y otras de similar entidad.

b) Es una afirmación falsa. Los módulos de E/S suministran una visión de los periférico en la que, desde un perspectiva Hw, basta con escribir (mandatos o datos) o leer (estado o datos también) en sus registros. En el caso particular de la técnica de E/S programada o directa bastaría con que los programas de manejo de los perféricos –los llamados *drivers*– tuviesen esta posibilidad de escritura y lectura en los registros, sin ningún mecanismo adicional por parte de la arquitectura. Estos registros se ven desde un punto de la CPU como direcciones en las que leer o escribir exactamente de la misma forma que en las correspondientes a direcciones de memoria. Por ello, algunas arquitecturas suministran instrucciones especiales para leer, `INPUT`, y escribir, `OUTPUT`, pero otras, las que usan el llamado 'mapa común', utilizan las misma que para memoria `LOAD/STORE` o `MOVE`.

c) De nuevo se trata de una afirmación falsa. El tiempo total que tarda en realizarse una operación de E/S depende fundamentalmente del periférico, ya que los tiempos de inicio y final de la operación –y menos significativamente, los tiempo de llenado o vaciado de los bufferes extremos en el bloque–, que sí que dependerían de los MIPS de la CPU, apenas influyen. Los tiempos que dependen del periférico propiamente dicho, i.e., el tiempo de acceso (si lo hubiera) y el tiempo de transferencia de bloque (que depende exclusivamente de su tamaño y de la velocidad de transferencia propia del periférico) son los que determinan fundamentalmente el tiempo de la operación.

d) Otra vez es una afirmación falsa. Las técnicas de E/S afectan fundamentalmente a tiempo de CPU que es necesario invertir en la operación de E/S, no al tiempo total de ésta, tiempo que, como acabamos de comentar, depende sobre todo de características del periférico.

2 **[2 puntos]** A una CPU de 32 bits de ancho de palabra y 2000 MIPS de capacidad de ejecución se desea conectar un periférico P1 que tiene una velocidad de 40 MB/s y un registro de datos de 32 bits. Se supone un funcionamiento mediante interrupciones donde la Secuencia de Reconocimiento de Interrupción tiene una duración de 2 ns y su rutina de interrupción 20 ns. Calcule:

a) La frecuencia de interrupción del periférico.

b) La duración máxima posible de su rutina de interrupción.

c) El número máximo de unidades del periférico que podrían operar simultáneamente.

d) El porcentaje de CPU disponible para otros procesos durante la operación conjunta de dos periféricos.

SOLUCIÓN

a) El periférico solicitará una interrupción por cada palabra, puesto que su buffer es un único registro de 32 bits (4 B). Consecuentemente:

$1\ Int/buffer \times 1buffer/4B \times 40 \cdot 10^6 B/s = 10 \cdot 10^6 Int/s$

b) Para calcular la duración máxima de la rutina de interrupción debemos determinar primero el período de las interrupciones, i.e., el tiempo que transcurre entre interrupciones consecutivas. Este valor se obtiene inmediatamente del calculado en el apartado anterior, puesto que es su inversa:

$\frac{32bits}{40 \cdot 8 \cdot 10^6 bits/s} = 100ns$

La capacidad de ejecución de la CPU, expresada ns/inst, es 0,5 ns/inst, por lo que entre cada dos solicitudes de interrupción se ejecutan 200 instrucciones.

De este período de las solicitudes de interrupción, la Secuencia de Reconocimiento de Interrupción consume 2 ns, por lo que quedarían un máximo teórico de 98 ns (196 instrucciones) para la rutina de interrupción de un único periférico.

c) De la frecuencia de solicitud de interrupción antes calculada y del consumo de CPU de cada una, t_{Int}, permitirá calcular el consumo máximo de cada unidad del periférico. De este consumo por unidad se obtendrá cómo se reparte el total de MIPS de la CPU.

Cada interrupción consume:

$t_{Int} = t_{SRI} + t_{RutInt} = 2ns + 20ns = 22ns$

y por lo tanto, el consumo de CPU por cada unidad de periférico será:

$ConsumoCPU/unidad = 10 \cdot 10^6 Int/s \times 22ns/Int \times 2inst/ns = 440MIPS/unidad$

por lo que los 2000 MIPS de la CPU podrían permitir el funcionamiento simultáneo de:

$\frac{2000MIPS}{440MIPS/unidad} = 4,45unidades \rightarrow 4unidades$

d) Cada unidad consume 440 MIPS, según acabamos de calcular, por lo que los 880 MIPS que representarían el funcionamiento simultáneo de dos unidades son un 44 % de consumo –y 56 % libre– sobre los 2000 MIPS de capacidad total de la CPU.

3 **[6 puntos]** Sean dos periféricos, un P1 que transfiere datos a una velocidad de 40 MB/s, que tiene un tiempo de acceso de 5 ms, bloques de 2 KB y un buffer de 16 registros de datos de 32 bits, y P2, que transfiere datos a 10^9 bits/s y que tiene un único registro de datos de 32 bits, ambos conectados a un procesador de 32 bits capaz de ejecutar 2000 MIPS y cuya SRI dura 2 ns.

Se conoce los siguientes datos sobre el código que se emplea para gestionar estos dos periféricos:

- P1: opera mediante interrupciones. Rutina de programación: 50 instrucciones; Rutina de interrupción: 40 instrucciones, con otras 20 adicionales al finalizar la operación. Bloques de 2 KB.
- P2: opera por DMA. Rutina de programación: 40 instrucciones; Rutina de interrupción: 50 instrucciones. En el DMA tarda 2 ns el protocolo de concesión y devolución de buses y 10 ns el acceso a memoria por cada palabra. Bloques de 0,5 KB.

a) Calcule la duración de una operación de E/S para el P1 y para el P2.

b) Calcule el porcentaje de CPU que queda libre para otros procesos durantes una operación de P1 y durante una operación de P2.

c) Suponga una operación en la que se lee un archivo de 10 KB desde P1 y se transmite por P2. Calcule:

- La duración total de la operación.
- El porcentaje de CPU disponible para otros procesos durante dicha operación.

d) Suponga ahora que al periférico P2 se le añade un buffer de 8 registros de 32 bits. Indique en qué medida esta modificación afectaría al tiempo de operación y al tiempo de CPU disponible para otros procesos.

SOLUCIÓN

a) El tiempo total de cada operación se obtiene de la suma de los tiempos e inicio o programación, el de acceso (si lo hubiera), el llamado tiempo de transferencia (i.e., lo que tarda el periférico en transmitir el bloque a su velocidad de transferencia), y por último el que se invierta en finalizar la operación:

$t_{op} = t_I + t_{acc} + t_{transf} + t_F$

P1:

$t_{op} = 25ns + 5 \cdot 10^6 ns + 51.200ns + 2ns + 20ns + 10ns = 5.051.247ns$

donde el tiempo de transferencia viene dado por la velocidad de P1:

$t_{transf} = \frac{2048B}{40 \cdot 10^6 B/s} = 51.200ns$

P2:

$t_{op} = 20ns + 0ms + 4.096ns + 2ns + 25ns = 4.143ns$

donde el tiempo de transferencia viene dado en este caso por la velocidad de P2:

$\frac{512B \times 8}{1 \cdot 10^9 bits/s} = 4.096ns$

b) El tiempo que está ocupada la CPU durante cada operación de E/S mediante interrupciones es fundamentalmente el que consumen las interrupciones, más los que se invierten en su programación o inicio y finalización, en este caso incluida en la última interrupción:

$t_{CPU} = t_I + n^o Int \times t_{Int} + t_F$

donde cada interrupción consume el tiempo de la SRI más el de la rutina de interrupción propiamente dicha:

$t_{Int} = t_{SRI} + t_{RutInt}$

En el caso de P1 solicita una interrupción por cada buffer de 16 palabras o 64 B:

$t_{CPU} = 25ns + \frac{2048B}{64B/Int} \times [2ns + 20ns]/Int + 10ns = 739ns$

P2 opera, por otro lado, mediante DMA, por lo que el consumo de CPU se deberá a las fases de inicio y finalización –que en este caso se supone asociada a la única interrupción– y al que corresponde a los DMAs, instantes en los que consideramos que la CPU deja de ejecutar al ceder el uso de los buses:

$t_{CPU} = t_I + n^o DMA \times t_{DMA} + t_F$

donde cada DMA consume el tiempo fijo del protocolo de concesión/liberación del bus más el de tantos accesos a memoria como palabras tenga en el buffer, uno solo en este caso, puesto que sólo dispone de un registro de datos:

$t_{DMA} = t_{conc/lib} + t_{Mp} = 2ns + 10ns = 12ns$

$t_{CPU} = 20ns + \frac{512B}{4B/DMA} \times [2ns + 10ns]/DMA + 2ns + 25ns = 1.583ns$

Los porcentajes de CPU libres durante toda la operación serán por tanto:

$\%CPUlibre = \frac{t_{op} - t_{CPU}}{t_{op}} \times 100$

P1:

$\%CPUlibre = \frac{5.051.147ns - 739ns}{5.051.147ns} \times 100 = 99,98\,\%$

P2:

$\%CPUlibre = \frac{4.143ns - 1.583ns}{4.143ns} \times 100 = 61,79\,\%$

c) Los 10 KB del archivo corresponderán a cinco operaciones de P1 (2 KB/operación) y veinte de P2 (0,5 KB/operación). Se comprueba fácilmente que es viable el funcionamiento simultáneo de ambos periféricos, por lo que cada operación de P1 –que son necesariamente anteriores y también mucho más lentas– vendría seguida de cuatro de P2. Consecuentemente, la duración total sería cinco tiempos de operación de P1 y los cuatro tiempos de operación de P2 correspondientes a los últimos 2 KB del archivo.

$t_{op10KBP1-->P2} = 5 \times t_{opP1} + 4 \times t_{opP2} = 5 \times 5.051.147ns + 4 \times 4.143ns = 25.272.307ns$

y el total de CPU que se invierte corresponderá a las diez operaciones de P1 y las veinte de P2:

$t_{CPU10KBP1-->P2} = 5 \times t_{CPUP1/op} + 20 \times t_{CPUP2/op} = 35.355ns$

por lo que queda el siguiente porcentaje de CPU libre durante la operación completa:

$\%CPUlibre = \frac{25.272.307ns - 19.995ns}{25.272.307ns} \times 100 = 99,86\,\%$

d) El uso de un buffer de 8 palabras en P2 reduciría el número de DMAs de 128 a 16. Ahora el coste en tiempo en cada DMA sería de un tiempo del protocolo de concesión/liberación y de ocho tiempos de acceso a memoria

en lugar de uno sólo. Por tanto, el consumo de CPU –con las consideraciones antes establecidas– soló se vería afectado en un cambio de 128 a 16 tiempos del protocolo de concesión/liberación del bus, i.e., 256 ns vs. 32 ns. Este ahorro sería de: 256 ns - 32 ns = 224 ns que habría que restarle al tiempo de ocupación en la configuración original: 1.583 ns vs. 1.359 ns (1.583 ns - 224 ns). La duración de cada operación de P2 tal y como la hemos calculado no se vería afectada por este cambio.

1 *Describa las ventajas e inconvenientes de una memoria principal con entrelazado simple de orden inferior frente a una memoria principal sin entrelazado.*

SOLUCIÓN

La memoria con entrelazado proporciona un ancho de banda teórico N veces superior a la memoria sin entrelazado cuando se accede a palabras consecutivas, siendo N (una potencia de 2) el número de módulos de memoria. Tal es el caso de las peticiones de las unidades de control de memorias caché que solicitan accesos a memoria para leer o escribir (en el caso de *Copy-Back*) bloques o líneas de caché.

Un inconveniente es que requieren más hardware para realizar estos accesos. En particular un registro de datos para cada uno de los módulos donde se almacena el bloque de palabras que se han leído o se van a escribir. En general, para escribir o leer estos registros se usa un bus con capacidad de transferencia de bloque, lo que reduce ligeramente el ancho de banda teórico, al necesitar un ciclo de reloj adicional para transmitir cada una de las palabras que componen el bloque. Sin embargo, el calificativo *simple* que acompaña al sustantivo entrelazado sugiere que este hardware adicional no es complejo.

2 *Explique la diferencia entre tiempo de acceso y tiempo de ocupación en los accesos a un sistema de memoria.*

SOLUCIÓN

El tiempo de acceso es el tiempo transcurrido desde que el procesador realiza una petición al sistema de memoria hasta que puede continuar, por ejemplo porque obtiene la información solicitada. El tiempo de ocupación es el tiempo transcurrido desde que el sistema de memoria recibe una petición hasta que la completa y puede servir la siguiente. A efectos prácticos, es el tiempo que está ocupado el sistema de memoria debido a una petición.

Las diferencias entre ambos tiempos se ponen claramente de manifiesto en memorias caché con alguna optimización. Por ejemplo, si la política de actualización es *Write-Through* y se dispone de buffer de escritura, el procesador podrá continuar con la ejecución de instrucciones tras escribir en el buffer, pero una petición inmediatamente posterior tendrá que esperar que el buffer se copie a memoria y se libere.

3 *Se tiene un computador de 32 bits que dispone de cachés separadas para instrucciones y datos con las siguientes características:*

- *Capacidad de cada memoria caché: 8 KB*
- *Tamaño de los bloques de caché: 16 bytes*
- *Organización asociativa por conjuntos de 2 bloques*
- *Política de reemplazo LRU* (Least Recently Used)
- *Política de lectura: OOF* (Out of Order Fetch)
- *Política de escritura de la caché de datos: diferida con actualización* (CBWA: Copy Back With Allocation)

Se está ejecutando en este computador el siguiente fragmento de un programa:

```
for (i=0; i<1024; i++)      /* 1.024 iteraciones */
   c[i] = a[i] + b[i];
```

Cada elemento de los vectores `a` *y* `b` *ocupa una palabra y están ubicados en las direcciones* `Df(a) = 00001000H` *y* `Df(b) = 00008800H`.

a) *Calcule en qué conjuntos de la caché de datos se ubicarán los dos primeros y los dos últimos bloques de los vectores* `a` *y* `b`.

b) *Si cada elemento del vector* `c` *también ocupa una palabra y el índice del bucle* `i` *se aloja en un registro del procesador, en qué direcciones ubicaría el vector* `c` *para evitar conflictos en la caché de datos.*

c) Calcule la tasa de aciertos de la caché de datos para el fragmento mostrado sabiendo que inicialmente se encuentra vacía.

d) Ubique el vector `c` *a partir de la posición de memoria que considere más conveniente y calcule el porcentaje de bloques sustituidos que han sido modificados al ejecutar este código.*

e) Calcule el tiempo medio de acceso al sistema de memoria durante la ejecución de este código, si el tiempo de acceso a la caché es de 2 ns, el tiempo de acceso a memoria principal 40 ns y el tiempo necesario para leer o escribir un bloque de 4 palabras 60 ns.[1]

SOLUCIÓN

a) Para resolver este apartado hay que calcular cómo se interpretan las direcciones físicas por la memoria caché. Como es asociativa por conjuntos habrá que calcular el número de conjuntos.

$$Núm\_conjuntos = \frac{8\ KB}{2\ Bq/Conjunto \cdot 16\ bytes/Bq} = 256\ Conjuntos$$

Por lo tanto las direcciones físicas de los dos primeros bloques de `a` y `b` se interpretan:

31 12	11 4	3 0	
Etiqueta	Conjunto	Byte	
00001	00	0	`Df(a)`
00008	80	0	`Df(b)`

Para los dos últimos bloques hay que calcular cuántos bloques ocupan los vectores `a` y `b`:

$$Núm\_bloques = \frac{1.024 \cdot 4\ bytes}{16\ bytes/Bq} = 256\ bloques$$

El número de bloques coincide con el número de conjuntos de la memoria caché. Por lo tanto, si el primer bloque del vector `a` va al conjunto 0, el último irá al conjunto 255. De forma similar, el primer bloque del vector `b` va al conjunto 128 (0x80) y el último al conjunto 127.

b) El vector `c` puede ubicarse en cualquier otra parte de la memoria y no se producirán conflictos en la memoria caché. Nótese que es asociativa por conjuntos de 2 bloques y que nunca se accede simultáneamente a un bloque del vector `a` y a uno del vector `b` a los que les corresponda el mismo conjunto.

c) Los únicos fallos que se producen son los forzosos o de primer acceso. El número total de bloques a los que se accede son los correspondientes a los 3 vectores, así tendremos $3 \cdot 256 = 768$ accesos con fallos.

$$M_r = \frac{Núm\_fallos}{Núm\_accessos} = \frac{768}{1.024 \cdot 3} = 0,25 \rightarrow H_r = 0,75$$

d) Al final de la ejecución del fragmento de código queda la caché llena, ya que los bloques de los 3 vectores se reparten por los 256 conjuntos. Como hemos visto, se cargan en caché 768 bloques y únicamente hay fallos forzosos. Comoquiera que la caché tiene sólo 512 bloques, se sustituyen 256 bloques.

Los únicos bloques que se modifican corresponden al vector `c`. Por lo tanto, si se coloca el vector `c` de tal manera que no se desaloje ninguno de sus bloques durante la ejecución del fragmento de código, el número de bloques que se sustituyen estando modificados será 0.

Según lo expuesto en el apartado b), el vector `c` se puede ubicar en cualquier parte de la memoria libre. Además, hay que tener en cuenta que en cada iteración del bucle se accede a los elementos del vector `c` en último lugar y que la política de reemplazo es LRU. Entonces, si el vector `c` se ubica en memoria de forma que a su primer bloque le corresponda el mismo conjunto que al primer bloque de `a` o `b`, se desalojarán éstos y nunca los de `c` y el porcentaje de bloques sustituidos que se han modificado es 0. Como ejemplo, se puede ubicar a partir de `Df(c) = 00000000H`, justo antes del vector `a`, o a partir de `Df(c) = 00009800H`, a continuación del vector `b`.

[1] Si no ha resuelto los apartados c) y d) considere una tasa de aciertos de 0,75 y una probabilidad de sustituir un bloque modificado de 0,25.

e) Para calcular este tiempo medio de acceso se tienen en cuenta los valores calculados en los apartados c) y d), con lo que $t_{penalización}$ es únicamente el tiempo de leer la palabra de memoria principal, por usar OOF y ser nula la probabilidad de sustituir un bloque modificado.

$$\bar{t}_{acceso} = t_{caché} + M_r \times t_{penalización} = 2\ ns + 0,25 \times 40\ ns = 12\ ns$$

4 *Considere ahora que al mismo computador del ejercicio 3 se le quiere dotar de memoria virtual paginada con dos niveles de tablas de páginas y sendas TLBs para la traducción de direcciones de datos e instrucciones.*

a) Calcule el tamaño mínimo de las páginas para que se pueda solapar la traducción de direcciones en las TLBs con el acceso a las memorias caché.

b) Indique cómo se interpretan las direcciones virtuales de 32 bits si las entradas de las tablas de páginas ocupan una palabra y cada tabla de páginas de primer nivel ocupa una página.

c) Calcule el tiempo máximo de acceso a este sistema de memoria teniendo en cuenta que el tiempo de acceso a las TLBs es el mismo que a las memorias caché y que no se producen fallos de página.

SOLUCIÓN

a) Para que se puedan solapar los accesos a las TLBs y memorias caché, basta con que el campo desplazamiento, que no se traduce, sea igual o mayor que los campos conjunto y byte.

Es decir necesitaríamos páginas de un tamaño mínimo de 2^{12} $bytes = 4\ KB$.

b) Si las páginas son de 4 KB y las entradas de las tablas de páginas son de una palabra (4 B), las tablas de página de primer nivel tendrán 1.024 entradas. De este modo las direcciones virtuales se interpretan así:

31 22	21 12	11 0
Entrada PVN1	Entrada PVN2	Desplazamiento

Donde todas las tablas de páginas, de primer y segundo nivel, tienen el tamaño de una página. Para páginas de mayor tamaño, la tabla de páginas de primer resultaría al menos 4 veces mayor que las de segundo nivel.

c) Para calcular el tiempo máximo de acceso, se calculará el tiempo máximo de traducción y el tiempo máximo de acceso a la información.

El tiempo máximo de traducción se produce cuando hay un fallo en la TLB y hay que traducir en los 2 niveles de tablas de páginas de memoria principal.

$$t_{máximo\_traducción} = 2\ ns + 2 \times 40\ ns = 82\ ns$$

El tiempo máximo de acceso a la información se produce al acceder a datos, cuando hay fallo en la memoria caché y el bloque a desalojar está modificado:

$$t_{máximo\_información} = 2\ ns + 60\ ns + 40\ ns = 102\ ns$$

Así se obtiene:

$$t_{máximo\_acceso} = 82\ ns + 102\ ns = 184\ ns$$